FLUCTUATING HYDRODYNAMICS ON GPUS

Compile

Edit the two firsts lines in the makefile, and type make. Note, this code needs the "HydroGrid" code from Donev to calculate the structure factors.

How to run

Type the executable name following with the input file

mainstandard inputfile

If a input file is not provided the code will search the file data.main.

Input File

The input file (data.main by default) contains the parameters for the simulation. The program doesn't use internal units, if the parameters are coherent the results should be fine. The input file should/might contain the following:
• Comments: start with the character #.

• Scheme: the code supports several schemes, chose only one between
RK3: RK3, without thermostat or concentration. Use
#nothing because RK3 is the default scheme

thermostat: RK3 with thermostat, without concentration. Use
thermostat

ghost: RK3 without thermostat or concentration. It's implemented using ghost cells and it's slower than the scheme RK3. Use
ghost

binaryMixture: RK3 with thermostat and concentration. It's implemented using ghost cells. Use
binaryMixture

binaryMixtureWall: like binaryMixture but with a hard wall in the planes y=0 and y=ly. Use
binaryMixtureWall

giantFluctuations: it's like binaryMixtureWall but with a different boundary conditions and the term $\nabla(\rho D c S_T \nabla T)$ in the concentration equation.
giantFluctuations

continuousGradient: it's like binaryMixture but with the additional term $\frac{1}{2}(v^y_{\boldsymbol{i}+\boldsymbol{y}/2} + v^y_{\boldsymbol{i}-\boldsymbol{y}/2})\rho_{bdi}\overline{\nabla c}$ in the equation for the density of the species 1. Provide the constant gradient in the input file with the parameter *gradTemperature*.
continuousGradient

• Fluid density
densfluid 0.632

• shearviscosity
shearviscosity 53.71

• bulkviscosity. Note that the term in grad(div(v)) is (bulkviscosity+shearviscosity/3)
bulkviscosity 127.05

• Pressure. In the code the pressure is a second order polynomial of the density $p = a0 + a1 * \rho + a2 * \rho^2$. Give the parameters a0, a1 and a2
#pressureparameters a0 a1 a2
pressureparameters 231.091 -947.837 920.671

• Temperature. The temperature is in fact (k_BT) with the appropiate units of energy
temperature 249.4

• Number of time steps
numsteps 1000

• Number of time step in which the code won't save data.
numstepsRelaxation 0

• Time step
dt 3.408

• Sample frequency
samplefreq 1

• Initialize fluid
initfluid option
with
option=0: fluid at rest, vx=vy=vz=0 and $\rho = cte$
option=1: close to thermal equilibrium, $< v_i(\mathbf{r}) >= 0$, $< v_i(\mathbf{r})v_j(\mathbf{s}) >= (k_BT/\rho V_{cell})\delta_{ij}\delta_{rs}$ and $\rho = cte$
In the simulation with concentration this initializes like $< concentration(y) >_{thermalequilibrium}$

• Output name. The output files created by the code start with "outputname". It's possible indicate the directory
outputname ../data/run1

• Number of cells mx my mz
cells 32 32 32

• Size system lx ly lz
celldimension 8000 8000 8000

• Random number seed. Default seed=time(0), that means that the code takes a seed from the computer's clock, but you can provide your own seed
seed 4

• Set device, In general it's better don't use this option.
setDevice numberDevice
setDevice 1

• Background velocity
backgroundvelocity vx0 vy0 vz0

The following options are only relevant for the binaryMixture or binaryMixtureWall schemes.

• Diffusion. For a ideal gas the diffusion is diffusion=A/ρ. Here you have to provide A
diffusion 3.66

• Mass species 0
massSpecies0 18

• Mass species 1
massSpecies1 18

• Concentration for the species 1
concentration 0.5

The following options are only relevant for the binaryMixtureWall scheme.
• Concentration in the walls y=0 and y=ly
concentrationWall 0.55 0.45

• Velocities in the walls y=0 and y=ly
vxWall 0 0

vzWall 0 0

The following options are only relevant for the giantFluctuations scheme.
• Soret coefficient
soretCoefficient 0.06486

• Temperature gradient
gradTemperature 174

Tips to modify the code

• Use the cuda compiler "nvcc".

• Take a look at the file main.cpp and follow the flow for one scheme, in the most part they are similar.

• The variables (i.e. vx, vy...) are duplicate, one copy resides in the CPU and the other in the GPU. They are not synchronize, you have to copy explicitly the variable from the CPU to the GPU or viceversa. The copy process is slow try to avoid copies.

• If different cuda functions ("kernels") use the same variable they and the variable must be declare in the same file. Take a look at the file GPU.cu.